L 5

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
```

```
**FILE**ID**QUORUM                                             F 12









  QQQQQQ    UU      UU    OOOOOO    RRRRRRRR    UU      UU  MM      MM
  QQQQQQ    UU      UU    OOOOOO    RRRRRRRR    UU      UU  MM      MM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MMMM  MMMM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MMMM  MMMM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MM  MM  MM
QQ      QQ  UU      UU  OO      OO  RR      RR  UU      UU  MM  MM  MM
QQ      QQ  UU      UU  OO      OO  RRRRRRRR    UU      UU  MM      MM
QQ      QQ  UU      UU  OO      OO  RRRRRRRR    UU      UU  MM      MM
QQ  QQ  QQ  UU      UU  OO      OO  RR  RR      UU      UU  MM      MM
QQ  QQ  QQ  UU      UU  OO      OO  RR  RR      UU      UU  MM      MM
QQ    QQ    UU      UU  OO      OO  RR    RR    UU      UU  MM      MM
QQ    QQ    UU      UU  OO      OO  RR    RR    UU      UU  MM      MM      ....
  QQQQ  QQ  UUUUUUUUUU    OOOOOO    RR      RR  UUUUUUUUUU  MM      MM      ....
  QQQQ  QQ  UUUUUUUUUU    OOOOOO    RR      RR  UUUUUUUUUU  MM      MM      ....
                                                                            ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
0000         .TITLE  QUORUM - DISK QUORUM MODULE
0000         .IDENT  'V04-000'
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
0000   ;++
0000   ; Facility: Executive, Cluster management
0000   ;
0000   ; Abstract:
0000   ;       This module contains the routines that implement the disk quorum
0000   ;       algorithm.
0000   ;
0000   ; Enviornment:
0000   ;       VMS Non Paged Exec - Kernel mode
0000   ;--
0000   ;
0000   ; Author:
0000   ;
0000   ;       R. Scott Hanna, CREATION DATE: 25-Jul-1983
0000   ;
0000   ; Modified by:
0000   ;
0000   ;       V03-008 WMC0003         Wayne Cardoza           16-Jul-1984
0000   ;               Call mount verification under some error conditions.
0000   ;               Clear CLUDCB$B_COUNTER on any entry to CLUSTER state.
0000   ;
0000   ;       V03-007 WMC0002         Wayne Cardoza           28-Jun-1984
0000   ;               Allow one error before calling CSP.
0000   ;
0000   ;       V03-006 WMC0001         Wayne Cardoza           31-May-1984
0000   ;               Make sure IRP$W_STS field is cleared.
0000   ;
0000   ;       V03-005 SSA0023         Stan Amway              6-Apr-1984
0000   ;               Decrement UCB device queue length when I/O completes
0000   ;               in READ_COMPLETE or WRITE_COMPLETE. This is required
```

```
0000    ;               because EXE$INSIOQ increments the length, but the IRP
0000    ;               does not go through the normal IOPOST code which does
0000    ;               the corresponding decrement.
0000    ;
0000    ;       V03-004 RSH0119         R. Scott Hanna          14-Mar-1984
0000    ;               Rewrite of module to use a new algorithm.
0000    ;
0000    ;       V03-003 RSH0085         R. Scott Hanna          23-Nov-1983
0000    ;               Remove clear of quorum file logical block number on
0000    ;               "connection" loss.
0000    ;
0000    ;       V03-002 RSH0078         R. Scott Hanna          10-Nov-1983
0000    ;               Changes in error handling to print error messages one
0000    ;               time only. Clear quorum file logical block number in
0000    ;               CLUDCB when "connection" is lost. Changes necessary due
0000    ;               to re-structured quorum block. Changes due to move of
0000    ;               QF_TRANS and QF_TIMEOUT from CLUB to CLUDCB.
0000    ;
0000    ;       V03-001 RSH0071         R. Scott Hanna          27-Sep-1983
0000    ;               Make sure CLUDCB$L_QBLAST and CLUDCB$L_QBBUF are
0000    ;               swapped on quorum file transition from inactive
0000    ;               regardless of the CLUB$V_QF_SKIP_READ bit.
0000    ;--
```

```
                                  0000            .SBTTL  Declarations
                                  0000    ;
                                  0000    ;       Define Symbols
                                  0000    ;
                                  0000
                                  0000            $CLUBDEF                                ; Cluster block
                                  0000            $CLUDCBDEF                              ; Cluster quorum disk control block
                                  0000            $CLUQFDEF                               ; Cluster quorum file
                                  0000            $CSBDEF                                 ; Cluster system block
                                  0000            $CSDDEF                                 ; Cluster server data
                                  0000            $CSPDEF                                 ; CSP communication codes
                                  0000            $DYNDEF                                 ; Dynamic data structure types
                                  0000            $IODEF                                  ; I/O function codes
                                  0000            $IPLDEF                                 ; Interrupt priority levels
                                  0000            $IRPDEF                                 ; I/O request packet
                                  0000            $SBDEF                                  ; System Block
                                  0000            $TQEDEF                                 ; Time queue entry
                                  0000            $UCBDEF                                 ; Unit control block
                                  0000            $VADEF                                  ; Virtual address fields
                                  0000
                                  0000    ;
                                  0000    ; The cycle count insures that we will not get burned by race conditions
                                  0000    ; and not see another cluster through the quorum disk.
                                  0000    ;
                        00000002  0000            CYCLE_COUNT = 2
                                  0000
                                  0000    ;
                                  0000    ; The following assumptions are in effect for the entire module
                                  0000    ;
                                  0000            ASSUME  IPL$_TIMER EQ IPL$_SYNCH
                                  0000            ASSUME  IPL$_TIMER EQ IPL$_SCS
                                  0000            ASSUME  CLUDCB$S_BUFFER EQ CLUQF$K_LENGTH
                                  0000            ASSUME  CLUQF$K_CHECK_LENGTH&3 EQ 0
                                  0000
                                  0000            .DEFAULT DISPLACEMENT,WORD
                                  0000
                                  0000    ;
                                  0000    ; Own Storage
                                  0000    ;
                                00000000          .PSECT  $$$060,LONG
                                  0000
                                  0000   CLUQF_IDENT_STRING:
45 4C 49 46 20 20 4D 55 52 4F 55 51  0000         .ASCII  /QUORUM  FILE/
                                  000C            ASSUME  CLUQF$S_IDENT EQ .-CLUQF_IDENT_STRING
                        000C     .SBTTL  CNX$QUORUM_INIT - Quorum initialization
                        000C
                        000C     ;++
                        000C     ; CNX$QUORUM_INIT - Quorum initialization
                        000C     ;
                        000C     ; FUNCTIONAL DESCRIPTION:
                        000C     ;
                        000C     ;       This routine determines if a quorum disk has been specified,
                        000C     ;       and if so allocates and initializes the cluster quorum disk
                        000C     ;       control block (CLUDCB) and associated data structures.
                        000C     ;
                        000C     ; CALLING SEQUENCE:
                        000C     ;
                        000C     ;       JSB/BSBx CNX$QUORUM_INIT
                        000C     ;           IPL is 31
                        000C     ;
                        000C     ; INPUTS:
                        000C     ;
                        000C     ;       NONE
                        000C     ;
                        000C     ; OUTPUT:
                        000C     ;
                        000C     ;       NONE
                        000C     ;
                        000C     ; SIDE EFFECTS:
                        000C     ;
                        000C     ;       R0-R5 are destroyed
                        000C     ;--
                        000C
                    00000000            .PSECT  $$$002,LONG                     ; Initialization PSECT
                        0000
                        0000     CNX$QUORUM_INIT::
                        0000
        0FC0 8F   BB    0000            PUSHR   #^M<R6,R7,R8,R9,R10,R11>        ; Save registers
                        0004     ;
                        0004     ; Determine if we have a quorum file
                        0004     ;
      10   20     3A    0004            LOCC    #^A/ /,#CLUDCB$S_DISK_QUORUM,-  ; Locate end of quorum disk name
   00000000'GF          0007                    G^CLU$GB_QDISK
      10   50     D1    000C            CMPL    R0,#CLUDCB$S_DISK_QUORUM        ; Is there a disk name?
           03     12    000F            BNEQU   1$                              ; Br if yes
         00A7     31    0011            BRW     4$
                        0014     ;
                        0014     ; Allocate the CLUDCB
                        0014     ;
51    00000229 8F   D0  0014     1$:    MOVL    #CLUDCB$K_LENGTH,R1             ; CLUDCB size
   00000000'GF    16    001B            JSB     G^EXE$ALONONPAGED               ; Allocate CLUDCB
        25 50     E9    0021            BLBC    R0,2$                           ; Br if error
      56   51     7D    0024            MOVQ    R1,R6                           ; Save CLUDCB size and address
                        0027     ;
                        0027     ; Allocate the IRP
                        0027     ;
51    000000C4 8F   D0  0027            MOVL    #IRP$K_LENGTH,R1                ; IRP size
   00000000'GF    16    002E            JSB     G^EXE$ALONONPAGED               ; Allocate IRP
        12 50     E9    0034            BLBC    R0,2$                           ; Br if error
      58   51     7D    0037            MOVQ    R1,R8                           ; Save IRP size and address
                        003A     ;
```

```
                                 003A      ; Allocate the TQE
                                 003A      ;
                51    30     D0  003A              MOVL    #TQE$K_LENGTH,R1                   ; TQE size
          00000000'GF        16  003D              JSB     G^EXE$ALONONPAGED                  ; Allocate TQE
                5A    51     7D  0043              MOVQ    R1,R10                             ; Save TQE size and address
                   03 50     E8  0046              BLBS    R0,3$                              ; Br if success
                    0076     31  0049      2$:     BRW     5$
                                 004C      ;
                                 004C      ; Initialize the CLUDCB
                                 004C      ;
  67    56    00    6E   00  2C  004C      3$:     MOVC5   #0,(SP),#0,R6,(R7)                 ; Zero the CLUDCB
                   08 A7  56 B0  0052              MOVW    R6,CLUDCB$W_SIZE(R7)               ; Store size
             0A A7    65 8F  90  0056              MOVB    #DYN$C_CLU,CLUDCB$B_TYPE(R7)       ; Store type
                         05  90  005B              MOVB    #DYN$C_CLU_CLUDCB,-                ; Store subtype
                      0B A7      005D                      CLUDCB$B_SUBTYPE(R7)
                   10 A7  59 D0  005F              MOVL    R9,CLUDCB$L_IRP(R7)                ; Store IRP address
                   14 A7  5B D0  0063              MOVL    R11,CLUDCB$L_TQE(R7)               ; Store TQE address
                         01  B0  0067              MOVW    #CLUDCB$M_QS_NOT_READY,-           ; Initial state is NOT_READY
                      20 A7      0069                      CLUDCB$W_STATE(R7)
                                 006B      ;
                                 006B      ; Initialize the IRP
                                 006B      ;
  69    58    00    6E   00  2C  006B              MOVC5   #0,(SP),#0,R8,(R9)                 ; Zero the IRP
                   08 A9  58 B0  0071              MOVW    R8,IRP$W_SIZE(R9)                  ; Store size
                   0A A9  0A 90  0075              MOVB    #DYN$C_IRP,IRP$B_TYPE(R9)          ; Store type
             23 A9    FF 8F  90  0079              MOVB    #^XFF,IRP$B_PRI(R9)                ; Store priority
                                 007E      ;
                                 007E      ; Initialize the TQE
                                 007E      ;
  6B    5A    00    6E   00  2C  007E              MOVC5   #0,(SP),#0,R10,(R11)               ; Zero the TQE
                   08 AB  5A D0  0084              MOVL    R10,TQE$W_SIZE(R11)                ; Store size
                   0A AB  0F 90  0088              MOVB    #DYN$C_TQE,TQE$B_TYPE(R11)         ; Store type
                   0B AB  05 90  008C              MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R11)    ; Store request type
          0C AB    0000'CF   9E  0090              MOVAB   QUORUM_TIMEOUT,TQE$L_FPC(R11)      ; Set up timer request fork PC
                   10 AB  57 D0  0096              MOVL    R7,TQE$L_FR3(R11)                  ; Store fork register three
        54    00000000'GF    D0  009A              MOVL    G^CLU$GL_CLUB,R4                   ; Get CLUB address
                   14 AB  54 D0  00A1              MOVL    R4,TQE$L_FR4(R11)                  ; Store fork register four
        52    00000000'GF    3C  00A5              MOVZWL  G^CLU$GW_QDSKINTERVAL,R2           ; Get timeout value. (in seconds)
  00    00989680 8F       52 7A  00AC              EMUL    R2,#10000000,#0,-                  ; Convert timeout to 100ns units
                      20 AB      00B4                      TQE$Q_DELTA(R11)                   ; ...and store in TQE
                                 00B6      ;
                                 00B6      ; Point CLUB to CLUDCB
                                 00B6      ;
              00B4 C4     57 D0  00B6              MOVL    R7,CLUB$L_CLUDCB(R4)               ; Store CLUDCB pointer in CLUB
                                 00BB
        50    00000000'8F    D0  00BB      4$:     MOVL    #SS$_NORMAL,R0                     ; Return success
                  0FC0 8F    BA  00C2      5$:     POPR    #^M<R6,R7,R8,R9,R10,R11>           ; Restore registers
                             05  00C6              RSB
                         00C7  .SBTTL  QUORUM_TIMEOUT - Quorum timeout
                         00C7  ;++
                         00C7  ; QUORUM_TIMEOUT - Quorum timeout
                         00C7  ;
                         00C7  ; FUNCTIONAL DESCRIPTION:
                         00C7  ;
                         00C7  ;       This routine executes every n seconds as a fork process where n is
                         00C7  ;       determined by the sysgen parameter QDSKINTERVAL.
                         00C7  ;
                         00C7  ; CALLING SEQUENCE:
                         00C7  ;
                         00C7  ;       JSB QUORUM_TIMEOUT
                         00C7  ;
                         00C7  ; INPUTS:
                         00C7  ;
                         00C7  ;       R3 = address of CLUDCB
                         00C7  ;       R4 = address of CLUB
                         00C7  ;       R5 = address of TQE
                         00C7  ;
                         00C7  ; OUTPUT:
                         00C7  ;
                         00C7  ;       R0-R2 Destroyed
                         00C7  ;--
                         00C7
                     00000000          .PSECT  $$$100,LONG
                         0000
                         0000  QUORUM_TIMEOUT::
                         0000
              56    DD   0000          PUSHL   R6                           ; Save R6
              00    E0   0002          BBS     #CLUDCB$V_QF_TIM,-           ; Br if we already timed out the
        2E 22 A3         0004                  CLUDCB$W_FLAGS(R3),5$        ; ...I/O in progress
     56    25 A3    DE   0007          MOVAL   CLUDCB$T_BUFFER(R3),R6       ; Get buffer address
              02    E1   000B          BBC     #CLUDCB$V_QF_WIP,-           ; Br if no write in progress
        05 22 A3         000D                  CLUDCB$W_FLAGS(R3),1$
           48 A6    96   0010          INCB    CLUQF$B_IGNORE(R6)           ; Invalidate buffer
              05    11   0013          BRB     2$
              01    E1   0015  1$:     BBC     #CLUDCB$V_QF_RIP,-           ; Br if no read in progress
        0E 22 A3         0017                  CLUDCB$W_FLAGS(R3),3$
              01    A8   001A  2$:     BISW2   #CLUDCB$M_QF_TIM,-           ; Set timeout bit
           22 A3         001C                  CLUDCB$W_FLAGS(R3)
     50     0000'CF 9E   001E          MOVAB   W^QDTIMOUT_MSG,R0            ; Point to timeout message
            03AF    30   0023          BSBW    QUORUM_DISK_TIMEOUT          ; Process timeout error
              0D    11   0026          BRB     5$
              00    E1   0028  3$:     BBC     #CLUDCB$V_QS_NOT_READY,-     ; Br if we are in one of the
        05 20 A3         002A                  CLUDCB$W_STATE(R3),4$        ; ...ready states
            03E4    30   002D          BSBW    REQUEST_CSP
              03    11   0030          BRB     5$
            0004    30   0032  4$:     BSBW    READ_QUORUM_FILE             ; Queue a quorum file read request
     56       8E    D0   0035  5$:     MOVL    (SP)+,R6                     ; Restore R6
              05         0038          RSB
```

```
                              0039  .SBTTL  READ_QUORUM_FILE - Queue a read to the quorum file
                              0039  ;++
                              0039  ; READ_QUORUM_FILE - Queue a read to the quorum file
                              0039  ;
                              0039  ; FUNCTIONAL DESCRIPTION:
                              0039  ;
                              0039  ;       This routine builds and queues an IRP to read the quorum file.
                              0039  ;
                              0039  ; CALLING SEQUENCE:
                              0039  ;
                              0039  ;       JSB/BSBx READ_QUORUM_FILE
                              0039  ;
                              0039  ; INPUTS:
                              0039  ;
                              0039  ;       R3 = address of CLUDCB
                              0039  ;       R6 = address of quorum file buffer
                              0039  ;
                              0039  ; OUTPUT:
                              0039  ;
                              0039  ;       R0-R2 destroyed
                              0039  ;--
                              0039
                              0039  READ_QUORUM_FILE:
                              0039
                   38    BB   0039          PUSHR   #^M<R3,R4,R5>                   ; Save registers
                   02    A8   003B          BISW    #CLUDCB$M_QF_RIP,-              ; Set read in progress bit
                22 A3         003D                  CLUDCB$W_FLAGS(R3)
          52    10 A3    D0   003F          MOVL    CLUDCB$L_IRP(R3),R2             ; Get IRP address
     0C A2    0097'CF    DE   0043          MOVAL   READ_COMPLETE,IRP$L_PID(R2)     ; Store completion address in PID
          55    0C A3    D0   0049          MOVL    CLUDCB$L_UCB(R3),R5             ; Get UCB address
          1C A2    55    D0   004D          MOVL    R5,IRP$L_UCB(R2)                ; Store UCB address
          20 A2    0C    B0   0051          MOVW    #IO$_READPBLK,IRP$W_FUNC(R2)    ; Store function code
                2A A2    B4   0055          CLRW    IRP$W_STS(R2)                   ; Mount verification bit may be set
       06 68 A5    02    E0   0058          BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),1$ ; Br if logical I/O
     2A A2    0100 8F    B0   005D          MOVW    #IRP$M_PHYSIO,IRP$W_STS(R2)     ; Set physical I/O flag in IRP
     32 A2    0204 8F    3C   0063  1$:     MOVZWL  #CLUQF$K_LENGTH,IRP$L_BCNT(R2)  ; Store byte count
30 A2    56   FE00 8F    AB   0069          BICW3   #^C<VA$M_BYTE>,R6,-             ; Store buffer start byte offset
                              0070                  IRP$W_BOFF(R2)
   51    56    15  09    EF   0070          EXTZV   #VA$V_VPN,#VA$S_VPN,R6,R1       ; Get buffer virtual page number
     50     00000000'GF  D0   0075          MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT base address
        2C A2    6041    DE   007C          MOVAL   (R0)[R1],IRP$L_SVAPTE(R2)       ; Store PTE address
          50    1C A3    D0   0081          MOVL    CLUDCB$L_QFLBN(R3),R0           ; Get logical block number
                53 52    D0   0085          MOVL    R2,R3                           ; Set up IRP address
          00000000'GF    16   0088          JSB     G^IOC$CVTLOGPHY                 ; Convert LBN to PBN
          00000000'GF    16   008E          JSB     G^EXE$INSIOQ                    ; Queue the request
                   38    BA   0094          POPR    #^M<R3,R4,R5>                   ; Restore registers
                         05   0096          RSB
```

```
                          0097  .SBTTL  READ_COMPLETE - Quorum file read complete
                          0097  ;++
                          0097  ; READ_COMPLETE - Quorum file read complete
                          0097  ;
                          0097  ; FUNCTIONAL DESCRIPTION:
                          0097  ;
                          0097  ;       This routine is called when the quorum file read completes.
                          0097  ;
                          0097  ; CALLING SEQUENCE:
                          0097  ;
                          0097  ;       JSB READ_COMPLETE
                          0097  ;
                          0097  ;           Called as a fork process by IOCIOPOST at IPL$_IOPOST
                          0097  ;
                          0097  ; INPUTS:
                          0097  ;
                          0097  ;       R5 = address of IRP
                          0097  ;
                          0097  ; OUTPUT:
                          0097  ;
                          0097  ;       R0-R5 destroyed
                          0097  ;--
                          0097
                          0097  READ_COMPLETE::
        00C0 8F      BB   0097          PUSHR   #^M<R6,R7>                      ; Save registers
     54   1C A5      D0   009B          MOVL    IRP$L_UCB(R5),R4                ; Get UCB address
          6A A4      B7   009F          DECW    UCB$W_QLEN(R4)                  ; Decrement device queue length
                          00A2          SETIPL  #IPL$_TIMER                     ; Raise IPL
54   00000000'GF     D0   00A5          MOVL    G^CLU$GL_CLUB,R4                ; Get CLUB address
   53    00B4 C4     D0   00AC          MOVL    CLUB$L_CLUDCB(R4),R3            ; Get CLUDCB address
     56   25 A3      DE   00B1          MOVAL   CLUDCB$T_BUFFER(R3),R6          ; Get quorum file buffer
             02      AA   00B5          BICW2   #CLUDCB$M_QF_RIP,-              ; Clear read in progress bit
          22 A3           00B7                  CLUDCB$W_FLAGS(R3)
   50    0000'CF     9E   00B9          MOVAB   W^QDRDERROR_MSG,R0              ; Assume read error
             00      E5   00BE          BBCC    #CLUDCB$V_QF_TIM,-              ; Br if read has not timed out
       13 22 A3           00C0                  CLUDCB$W_FLAGS(R3),10$
       50 38 A5      E8   00C3          BLBS    IRP$L_IOST1(R5),40$             ; Br if read was successful
           035C      30   00C7          BSBW    CHECK_ERROR                     ; Is error fatal?
          4A 50      E8   00CA          BLBS    R0,40$                          ; Continue
             01      B0   00CD          MOVW    #CLUDCB$M_QS_NOT_READY,-        ; Set state to not ready
          20 A3           00CF                  CLUDCB$W_STATE(R3)
           0340      30   00D1          BSBW    REQUEST_CSP
             41      11   00D4          BRB     40$
       15 38 A5      E8   00D6  10$:    BLBS    IRP$L_IOST1(R5),14$             ; Br if no read error
           0349      30   00DA          BSBW    CHECK_ERROR                     ; Is error fatal?
          37 50      E8   00DD          BLBS    R0,40$                          ; Continue
   50    0000'CF     9E   00E0          MOVAB   W^QDRDERROR_MSG,R0              ; Read error
             05      E2   00E5          BBSS    #CLUDCB$V_QF_FIRST_ERR,-        ; Is this first error
       15 22 A3           00E7                  CLUDCB$W_FLAGS(R3),20$
           02E8      30   00EA          BSBW    QUORUM_FILE_RETRY               ; Process error
             28      11   00ED          BRB     40$
             05      E5   00EF  14$:    BBCC    #CLUDCB$V_QF_FIRST_ERR,-        ; Clear any previous error
       00 22 A3           00F1                  CLUDCB$W_FLAGS(R3),15$
           0260      30   00F4  15$:    BSBW    VALIDATE_QUORUM_FILE
          0A 50      E8   00F7          BLBS    R0,30$                          ; Br if quorum file valid
   50    0000'CF     9E   00FA          MOVAB   W^QDINVDAT_MSG,R0               ; Point to invalid data message
           02D8      30   00FF  20$:    BSBW    QUORUM_FILE_ERROR               ; Process error
```

```
                    13   11  0102      BRB     40$
           04       01   EA  0104 30$: FFS     #CLUDCB$V_QS_READY,#4,-          ; Get relative state bit position
       50      20   A3       0107              CLUDCB$W_STATE(R3),R0
     51    011F'CF       DE  010A      MOVAL   DISPATCH,R1                      ; Get dispatch table address
  51    011B'CF40        C0  010F      ADDL2   DISPATCH-4[R0],R1                ; Form routine address
                    61   16  0115      JSB     (R1)                             ; Dispatch to routine
                             0117 40$: SETIPL  #IPL$_IOPOST                     ; Restore IPL
           00C0     8F   BA  011A      POPR    #^M<R6,R7>                       ; Restore registers
                         05  011E      RSB
                             011F
                00000010'    011F DISPATCH:    .LONG   READ_COMPLETE_READY-DISPATCH
                0000004F'    0123              .LONG   READ_COMPLETE_ACTIVE-DISPATCH
                00000087'    0127              .LONG   READ_COMPLETE_CLUSTER-DISPATCH
                00000087'    012B              .LONG   READ_COMPLETE_VOTE-DISPATCH
```

```
                        012F  .SBTTL  READ_COMPLETE_READY - Read complete processing for READY state
                        012F  ;++
                        012F  ; READ_COMPLETE_READY - Read complete processing for READY state
                        012F  ;
                        012F  ; FUNCTIONAL DESCRIPTION:
                        012F  ;
                        012F  ;       This routine performs the read complete processing specific
                        012F  ;       to the READY state.
                        012F  ;
                        012F  ; CALLING SEQUENCE:
                        012F  ;
                        012F  ;       JSB/BSBx READ_COMPLETE_READY
                        012F  ;
                        012F  ; INPUTS:
                        012F  ;
                        012F  ;       R3 = address of CLUDCB
                        012F  ;       R4 = address of CLUB
                        012F  ;       R6 = address of quorum file buffer
                        012F  ;
                        012F  ; OUTPUT:
                        012F  ;
                        012F  ;       R0-R2,R5 Destroyed
                        012F  ;--
                        012F
                        012F  READ_COMPLETE_READY:
                        012F
              04   B0   012F          MOVW    #CLUDCB$M_QS_ACTIVE,-           ; Set state to active
           20 A3        0131                  CLUDCB$W_STATE(R3)
              08   AA   0133          BICW    #CLUDCB$M_QF_ERROR,-            ; Clear error reported bit
           22 A3        0135                  CLUDCB$W_FLAGS(R3)
         0200 C6   D0   0137          MOVL    CLUQF$L_ACT_COUNT(R6),-         ; Save activity longword
           18 A3        013B                  CLUDCB$L_ACT_COUNT(R3)
   00C8 C4    00   D2   013D          MCOML   #0,CLUB$L_FOREIGN_CLUSTER(R4)   ; Fill shift register with 1's
              02   C8   0142          BISL    #CLUB$M_QF_ACTIVE,-             ; Set active bit
           1C A4        0144                  CLUB$L_FLAGS(R4)
   50    0000'CF   9E   0146          MOVAB   W^QDCON_MSG,R0                  ; Point to connect message
              55   D4   014B          CLRL    R5                              ; No CSB
           FEB0'   30   014D          BSBW    CNX$CONFIG_CHANGE               ; Output message
           FEAD'   30   0150          BSBW    CNX$DISK_CHANGE                 ; Let connection manager know
              00   E1   0153          BBC     #CLUB$V_CLUSTER,-               ; Br if local node not a
        15 1C A4        0155                  CLUB$L_FLAGS(R4),1$             ; ...cluster member
              08   B0   0158          MOVW    #CLUDCB$M_QS_CLUSTER,-          ; Set state to cluster
           20 A3        015A                  CLUDCB$W_STATE(R3)
           24 A3   94   015C          CLRB    CLUDCB$B_COUNTER(R3)            ; Clear counter
     01000000 8F   CA   015F          BICL    #CLUB$M_QF_FAILED_NODE,-        ; Clear failout bit in CLUB
           1C A4        0165                  CLUB$L_FLAGS(R4)
            008B   30   0167          BSBW    BUILD_QUORUM_FILE               ; Build the owner & activity blocks
            00F2   30   016A          BSBW    WRITE_QUORUM_OWNACT             ; Write the owner & activity blocks
              05        016D  1$:     RSB
```

```
                        016E  .SBTTL  READ_COMPLETE_ACTIVE - Read complete processing for ACTIVE state
                        016E  ;++
                        016E  ; READ_COMPLETE_ACTIVE - Read complete processing for ACTIVE state
                        016E  ;
                        016E  ; FUNCTIONAL DESCRIPTION:
                        016E  ;
                        016E  ;       This routine performs the read complete processing specific
                        016E  ;       to the ACTIVE state.
                        016E  ;
                        016E  ; CALLING SEQUENCE:
                        016E  ;
                        016E  ;       JSB/BSBx READ_COMPLETE_ACTIVE
                        016E  ;
                        016E  ; INPUTS:
                        016E  ;
                        016E  ;       R3 = address of CLUDCB
                        016E  ;       R4 = address of CLUB
                        016E  ;       R6 = address of quorum file buffer
                        016E  ;
                        016E  ; OUTPUT:
                        016E  ;
                        016E  ;       R0-R2 Destroyed
                        016E  ;--
                        016E
                        016E  READ_COMPLETE_ACTIVE:
                        016E
            00     E1   016E          BBC     #CLUB$V_CLUSTER,-                 ; Br if local node not a
         17 1C A4       0170                  CLUB$L_FLAGS(R4),1$               ; ...cluster member
            08     B0   0173          MOVW    #CLUDCB$M_QS_CLUSTER,-            ; Set state to cluster
            20 A3       0175                  CLUDCB$W_STATE(R3)
         24 A3     94   0177          CLRB    CLUDCB$B_COUNTER(R3)              ; Clear counter
   01000000 8F     CA   017A          BICL    #CLUB$M_QF_FAILED_NODE,-          ; Clear failout bit in CLUB
            1C A4       0180                  CLUB$L_FLAGS(R4)
          0070     30   0182          BSBW    BUILD_QUORUM_FILE                 ; Build the owner & activity blocks
          00D7     30   0185          BSBW    WRITE_QUORUM_OWNACT               ; Write the owner & activity blocks
            1B     11   0188          BRB     2$
  00C8 C4   01     78   018A  1$:     ASHL    #1,CLUB$L_FOREIGN_CLUSTER(R4),- ; Assume no activity
       00C8 C4          018F                  CLUB$L_FOREIGN_CLUSTER(R4)
       0200 C6     D1   0192          CMPL    CLUQF$L_ACT_COUNT(R6),-           ; Activity longword change?
         18 A3          0196                  CLUDCB$L_ACT_COUNT(R3)
            0B     13   0198          BEQLU   2$                                ; Br if not
  00C8 C4   01     C8   019A          BISL    #1,CLUB$L_FOREIGN_CLUSTER(R4)     ; We have seen a foreign cluster
       0200 C6     D0   019F          MOVL    CLUQF$L_ACT_COUNT(R6),-           ; Save activity longword
         18 A3          01A3                  CLUDCB$L_ACT_COUNT(R3)
                   05   01A5  2$:     RSB
```

```
                          01A6  .SBTTL  READ_COMPLETE_CLUSTER/VOTE - Read complete processing for CLUSTER and VOTE s
                          01A6  ;++
                          01A6  ; READ_COMPLETE_CLUSTER - Read complete processing for CLUSTER state
                          01A6  ; READ_COMPLETE_VOTE - Read complete processing for VOTE state
                          01A6  ;
                          01A6  ; FUNCTIONAL DESCRIPTION:
                          01A6  ;
                          01A6  ;       This routine performs the read complete processing specific
                          01A6  ;       to the CLUSTER and VOTE states.
                          01A6  ;
                          01A6  ; CALLING SEQUENCE:
                          01A6  ;
                          01A6  ;       JSB/BSBx READ_COMPLETE_CLUSTER
                          01A6  ;       JSB/BSBx READ_COMPLETE_VOTE
                          01A6  ;
                          01A6  ; INPUTS:
                          01A6  ;
                          01A6  ;       R3 = address of CLUDCB
                          01A6  ;       R4 = address of CLUB
                          01A6  ;       R6 = address of quorum file buffer
                          01A6  ;
                          01A6  ; OUTPUT:
                          01A6  ;
                          01A6  ;       R0-R2,R5 Destroyed
                          01A6  ;--
                          01A6
                          01A6  READ_COMPLETE_CLUSTER:
                          01A6  READ_COMPLETE_VOTE:
                          01A6
            18    E5      01A6          BBCC    #CLUB$V_QF_FAILED_NODE,-        ; Br if node was not failed out
      06 1C A4            01A8                  CLUB$L_FLAGS(R4),1$
            08    B0      01AB          MOVW    #CLUDCB$M_QS_CLUSTER,-          ; Set state to CLUSTER
         20 A3            01AD                  CLUDCB$W_STATE(R3)
            3A    11      01AF          BRB     4$
         48 A6    95      01B1  1$:     TSTB    CLUQF$B_IGNORE(R6)              ; Is data in quorum file stale?
            35    12      01B4          BNEQU   4$                              ; Br if yes
          01C1    30      01B6          BSBW    CHECK_OWNER                     ; Determine who owns quorum file
         08 50    E9      01B9          BLBC    R0,2$                           ; Br if not a member of my cluster
         24 A3    96      01BC          INCB    CLUDCB$B_COUNTER(R3)            ; Increment counter
          00AA    30      01BF          BSBW    WRITE_QUORUM_ACT                ; Write the activity block
            30    11      01C2          BRB     5$
50     0000'CF    9E      01C4  2$:     MOVAB   W^QDFORCLUS_MSG,R0              ; Point to foreign cluster message
            55    D4      01C9          CLRL    R5                              ; No CSB
         FE32'    30      01CB          BSBW    CNX$CONFIG_CHANGE               ; Output message
            00    E0      01CE          BBS     #CLUQF$V_QUORUM,-               ; Bugcheck if he has dynamic quorum
      13 0E A6            01D0                  CLUQF$W_FLAGS(R6),3$
            1C    E0      01D3          BBS     #CLUB$V_QUORUM,-                ; Continue if we have dynamic quorum
      13 1C A4            01D5                  CLUB$L_FLAGS(R4),4$
         36 A6    B1      01D8          CMPW    CLUQF$W_VOTES(R6),-             ; Does he have static quorum?
         34 A6            01DB                  CLUQF$W_QUORUM(R6)
            07    1E      01DD          BGEQU   3$                              ; Br if yes
         22 A4    B1      01DF          CMPW    CLUB$W_VOTES(R4),-              ; Do we have static quorum?
         20 A4            01E2                  CLUB$W_QUORUM(R4)
            05    1E      01E4          BGEQU   4$                              ; Br if yes
         FE17'    30      01E6  3$:     BSBW    CNX$BUGCHECK_CLUSTER            ; Cause all nodes to bugcheck
            09    11      01E9          BRB     5$
         24 A3    94      01EB  4$:     CLRB    CLUDCB$B_COUNTER(R3)            ; Clear counter
```

```
                 0004   30  01EE    557          BSBW    BUILD_QUORUM_FILE                  ; Build the owner & activity blocks
                 006B   30  01F1    558          BSBW    WRITE_QUORUM_OWNACT                ; Write the owner & activity blocks
                        05  01F4    559 5$:      RSB
```

```
                          01F5   .SBTTL  BUILD_QUORUM_FILE - Build the quorum file owner and activity blocks
                          01F5   ;++
                          01F5   ; BUILD_QUORUM_FILE - Build the quorum file owner and activity blocks
                          01F5   ;
                          01F5   ; FUNCTIONAL DESCRIPTION:
                          01F5   ;
                          01F5   ;       This routine builds the quorum file owner and activity blocks.
                          01F5   ;
                          01F5   ; CALLING SEQUENCE:
                          01F5   ;
                          01F5   ;       JSB/BSBx BUILD_QUORUM_FILE
                          01F5   ;
                          01F5   ; INPUTS:
                          01F5   ;
                          01F5   ;       R4 = address of CLUB
                          01F5   ;       R6 = address of quorum file buffer
                          01F5   ;
                          01F5   ; OUTPUT:
                          01F5   ;
                          01F5   ;       R0-R2 destroyed
                          01F5   ;--
                          01F5
                          01F5   BUILD_QUORUM_FILE:
                          01F5
                          01F5           ASSUME  CLUQF$K_VERSION EQ 2            ; Assume version 2 structure
                          01F5           ASSUME  SB$S_SYSTEMID EQ 6              ; Assume system ID is 6 bytes
          00B8 8F    BB   01F5           PUSHR   #^M<R3,R4,R5,R7>                ; Save registers
            0C       28   01F9           MOVC3   #CLUQF$S_IDENT,-                ; Store ID string
      66  0000'CF         01FB                   CLUQF_IDENT_STRING,(R6)
            50  53   D0   01FF           MOVL    R3,R0                           ; R0 = current buffer pointer
            53  6E   7D   0202           MOVQ    (SP),R3                         ; Restore CLUDCB and CLUB pointers
            80  02   B0   0205           MOVW    #CLUQF$K_VERSION,(R0)+          ; Store QF version number
            80  01   B0   0208           MOVW    #CLUQF$M_QUORUM,(R0)+           ; Assume we have dynamic quorum
                1C   E0   020B           BBS     #CLUB$V_QUORUM,-                ; Br if we do have dynamic quorum
          03  1C A4       020D                   CLUB$L_FLAGS(R4),1$
             FE A0   B4   0210           CLRW    -2(R0)                          ; Fix the incorrect assumption
         80  2C A4   7D   0213   1$:     MOVQ    CLUB$Q_FTIME(R4),(R0)+          ; Store FOU_TIME
         80  3C A4   7D   0217           MOVQ    CLUB$Q_LST_TIME(R4),(R0)+       ; Store LST_TIME
  80   00000000'GF   7D   021B           MOVQ    G^EXE$GQ_SYSTIME,(R0)+          ; Store QF_TIME
  80   0000002C'GF   7D   0222           MOVQ    G^SCS$GA_LOCALSB+SB$Q_SWINCARN,(R0)+ ; Store SWINCARN
         80  60 A4   D0   0229           MOVL    CLUB$L_LOCAL_CSID(R4),(R0)+     ; Store CSID
         80  20 A4   B0   022D           MOVW    CLUB$W_QUORUM(R4),(R0)+         ; Store cluster quorum
         80  22 A4   B0   0231           MOVW    CLUB$W_VOTES(R4),(R0)+          ; Store cluster votes
  80   00000018'GF   D0   0235           MOVL    G^SCS$GA_LOCALSB+SB$B_SYSTEMID,(R0)+ ; Store system ID
  80   0000001C'GF   B0   023C           MOVW    G^SCS$GA_LOCALSB+SB$B_SYSTEMID+4,(R0)+
         80  26 A4   D0   0243           MOVL    CLUB$B_FSYSID(R4),(R0)+         ; Store FSYSID
         80  2A A4   B0   0247           MOVW    CLUB$B_FSYSID+4(R4),(R0)+
                60   D4   024B           CLRL    (R0)                            ; Initialize checksum
             04 A0   94   024D           CLRB    4(R0)                           ; Zero the ignore flag
              016F   30   0250           BSBW    CALCULATE_CHECKSUM              ; Calculate the owner block checksum
            60  57   D0   0253           MOVL    R7,(R0)                         ; Store checksum
          0200 C6    D6   0256           INCL    CLUQF$L_ACT_COUNT(R6)           ; Increment the activity counter
          00B8 8F    BA   025A           POPR    #^M<R3,R4,R5,R7>                ; Restore registers
                     05   025E           RSB
```

```
                              025F  .SBTTL  Quorum file write routines
                              025F  ;++
                              025F  ; WRITE_QUORUM_OWNACT - Write the quorum file owner and activity blocks
                              025F  ; WRITE_QUORUM_ACT - Write the quorum file activity block
                              025F  ;
                              025F  ; FUNCTIONAL DESCRIPTION:
                              025F  ;
                              025F  ;       This routine builds and queues an IRP to write the owner and activity
                              025F  ;       block or just the activity block to the quorum file.
                              025F  ;
                              025F  ; CALLING SEQUENCE:
                              025F  ;
                              025F  ;       JSB/BSBx WRITE_QUORUM_OWNACT
                              025F  ;       JSB/BSBx WRITE_QUORUM_ACT
                              025F  ;
                              025F  ; INPUTS:
                              025F  ;
                              025F  ;       R3 = address of CLUDCB
                              025F  ;       R6 = address of quorum file buffer
                              025F  ;
                              025F  ; OUTPUT:
                              025F  ;
                              025F  ;       R0-R2 destroyed
                              025F  ;--
                              025F          .ENABLE LSB
                              025F
                              025F  WRITE_QUORUM_OWNACT:
                              025F
           0078 8F     BB     025F          PUSHR   #^M<R3,R4,R5,R6>                ; Save registers
                7E     D4     0263          CLRL    -(SP)                           ; Quorum file block 0
     7E    0204 8F     3C     0265          MOVZWL  #CLUQF$K_LENGTH,-(SP)           ; Byte count
                11     11     026A          BRB     1$
                              026C
                              026C  WRITE_QUORUM_ACT:
                              026C
           0078 8F     BB     026C          PUSHR   #^M<R3,R4,R5,R6>                ; Save registers
     56    0200 C6     DE     0270          MOVAL   CLUQF$L_ACT_COUNT(R6),R6        ; Get activity block address
                66     D6     0275          INCL    (R6)                            ; Increment the activity counter
           7E   01     9A     0277          MOVZBL  #1,-(SP)                        ; Quorum file block 1
           7E   04     9A     027A          MOVZBL  #CLUQF$K_ACT_LENGTH,-(SP)       ; Byte count
                04     A8     027D  1$:     BISW    #CLUDCB$M_QF_WIP,-              ; Set write in progress bit
             22 A3            027F          CLUDCB$W_FLAGS(R3)
        52   10 A3     D0     0281          MOVL    CLUDCB$L_IRP(R3),R2             ; Get IRP address
   0C A2   02DA'CF     DE     0285          MOVAL   WRITE_COMPLETE,IRP$L_PID(R2)    ; Store completion address in PID
        55   0C A3     D0     028B          MOVL    CLUDCB$L_UCB(R3),R5             ; Get UCB address
        1C A2   55     D0     028F          MOVL    R5,IRP$L_UCB(R2)                ; Store UCB address
        20 A2   08     B0     0293          MOVW    #IO$_WRITEPBLK,IRP$W_FUNC(R2)   ; Store function code
             2A A2     B4     0297          CLRW    IRP$Q_STS(R2)                   ; Mount verification bit may be set
     06 68 A5   02     E0     029A          BBS     #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),2$ ; Br if logical I/O
   2A A2   0100 8F     B0     029F          MOVW    #IRP$M_PHYSIO,IRP$W_STS(R2)     ; Set physical I/O flag in IRP
        32 A2   8E     D0     02A5  2$:     MOVL    (SP)+,IRP$L_BCNT(R2)            ; Store byte count
30 A2   56 FE00 8F     AB     02A9          BICW3   #^C<VA$M_BYTE>,R6,-             ; Store buffer start byte offset
                              02B0          IRP$W_BOFF(R2)
  51   56   15  09     EF     02B0          EXTZV   #VA$V_VPN,#VA$S_VPN,R6,R1       ; Get buffer virtual page number
   50   00000000'GF    D0     02B5          MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT base address
     2C A2    6041     DE     02BC          MOVAL   (R0)[R1],IRP$L_SVAPTE(R2)       ; Store PTE address
  50    1C A3   8E     C1     02C1          ADDL3   (SP)+,CLUDCB$L_QFLBN(R3),R0     ; Get logical block number
```

```
         53    52  D0  02C6   673          MOVL    R2,R3                              ; Set up IRP address
    00000000'GF    16  02C9   674          JSB     G^IOC$CVTLOGPHY                    ; Convert LBN to PBN
    00000000'GF    16  02CF   675          JSB     G^EXE$INSIOQ                       ; Queue the request
       0078 8F     BA  02D5   676          POPR    #^M<R3,R4,R5,R6>                   ; Restore registers
                   05  02D9   677          RSB
                       02DA   678
                       02DA   679          .DISABLE LSB
```

```
                              02DA  .SBTTL  WRITE_COMPLETE - Quorum file write complete
                              02DA  ;++
                              02DA  ; WRITE_COMPLETE - Quorum file write complete
                              02DA  ;
                              02DA  ; FUNCTIONAL DESCRIPTION:
                              02DA  ;
                              02DA  ;       This routine is called when a quorum file write completes.
                              02DA  ;
                              02DA  ; CALLING SEQUENCE:
                              02DA  ;
                              02DA  ;       JSB WRITE_COMPLETE
                              02DA  ;
                              02DA  ;           Called as a fork process by IOCIOPOST at IPL$_IOPOST
                              02DA  ;
                              02DA  ; INPUTS:
                              02DA  ;
                              02DA  ;       R5 = address of IRP
                              02DA  ;
                              02DA  ; OUTPUT:
                              02DA  ;
                              02DA  ;       R0-R4 destroyed
                              02DA  ;--
                              02DA
                              02DA  WRITE_COMPLETE::
       54    1C A5     D0     02DA          MOVL    IRP$L_UCB(R5),R4                 ; Get UCB address
             6A A4     B7     02DE          DECW    UCB$W_QLEN(R4)                   ; Decrement device queue length
                              02E1          SETIPL  #IPL$_TIMER                      ; Raise IPL
  54   00000000'GF     D0     02E4          MOVL    G^CLU$GL_CLUB,R4                 ; Get CLUB address
     53    00B4 C4     D0     02EB          MOVL    CLUB$L_CLUDCB(R4),R3             ; Get CLUDCB address
                04     AA     02F0          BICW2   #CLUDCB$M_QF_WIP,-               ; Clear write in progress bit
             22 A3            02F2                  CLUDCB$W_FLAGS(R3)
     50    0000'CF     9E     02F4          MOVAB   W^QDWRERROR_MSG,R0               ; Point to write error message
                00     E5     02F9          BBCC    #CLUDCB$V_QF_TIM,-               ; Br if write has not timed out
          13 22 A3            02FB                  CLUDCB$W_FLAGS(R3),10$
          51 38 A5     E8     02FE          BLBS    IRP$L_IOST1(R5),30$              ; Br if write was successful
              0121     30     0302          BSBW    CHECK_ERROR                      ; Is error fatal?
             4B 50     E8     0305          BLBS    R0,30$                           ; Continue
                01     B0     0308          MOVW    #CLUDCB$M_QS_NOT_READY,-         ; Set state to not ready
             20 A3            030A                  CLUDCB$W_STATE(R3)
              0105     30     030C          BSBW    REQUEST_CSP                      ; Request the CSP process
                42     11     030F          BRB     30$
          1A 38 A5     E8     0311  10$:    BLBS    IRP$L_IOST1(R5),20$              ; Br if write success
              010E     30     0315          BSBW    CHECK_ERROR                      ; Is error fatal?
             38 50     E8     0318          BLBS    R0,30$                           ; Continue
     50    0000'CF     9E     031B          MOVAB   W^QDWRERROR_MSG,R0               ; Point to write error message
                05     E2     0320          BBSS    #CLUDCB$V_QF_FIRST_ERR,-         ; Is this first error
          05 22 A3            0322                  CLUDCB$W_FLAGS(R3),15$
              00AD     30     0325          BSBW    QUORUM_FILE_RETRY                ; Process error (retry)
                29     11     0328          BRB     30$
              00AD     30     032A  15$:    BSBW    QUORUM_FILE_ERROR                ; Process error
                24     11     032D          BRB     30$
                              032F  20$:    ASSUME  CLUDCB$M_QF_WRL_ERR LE 255
             40 8F     8A     032F          BICB    #CLUDCB$M_QF_WRL_ERR,-           ; Not write locked
             22 A3            0332                  CLUDCB$W_FLAGS(R3)
                04     E0     0334          BBS     #CLUDCB$V_QS_VOTE,-              ; Br if state = VOTE
          1A 20 A3            0336                  CLUDCB$W_STATE(R3),30$
                02     91     0339          CMPB    #CYCLE_COUNT,-                   ; Have we cycled enough?
          24 A3       033B   738              CLUDCB$B_COUNTER(R3)
             14   12  033D   739         BNEQU  30$                          ; Br if not
             18   E0  033F   740         BBS    #CLUB$V_QF_FAILED_NODE,-     ; Br if a node has been failed out
       0F 1C A4       0341   741                CLUB$L_FLAGS(R4),30$
             10   B0  0344   742         MOVW   #CLUDCB$M_QS_VOTE,-          ; Set state to VOTE
          20 A3       0346   743                CLUDCB$W_STATE(R3)
   40000000 8F    C8  0348   744         BISL   #CLUB$M_QF_DYNVOTE,-         ; Set dynamic vote bit in CLUB
          1C A4       034E   745                CLUB$L_FLAGS(R4)
          FCAD'   30  0350   746         BSBW   CNX$DISK_CHANGE              ; Let connection manager know
                      0353   747 30$:    SETIPL #IPL$_IOPOST                 ; Restore IPL
                  05  0356   748         RSB
```

```
                         0357  .SBTTL  VALIDATE_QUORUM_FILE - Validate quorum file
                         0357  ;++
                         0357  ; VALIDATE_QUORUM_FILE - Validate quorum file
                         0357  ;
                         0357  ; FUNCTIONAL DESCRIPTION:
                         0357  ;
                         0357  ;       This routine validates the contents of the quorum file.
                         0357  ;
                         0357  ; CALLING SEQUENCE:
                         0357  ;
                         0357  ;       JSB/BSBx VALIDATE_QUORUM_FILE
                         0357  ;
                         0357  ; INPUTS:
                         0357  ;
                         0357  ;       R6 = address of quorum file buffer
                         0357  ;
                         0357  ; OUTPUT:
                         0357  ;
                         0357  ;       R0 = status
                         0357  ;           0 - The block is invalid
                         0357  ;           1 - The block is valid
                         0357  ;
                         0357  ;       R1-R2 destroyed
                         0357  ;--
                         0357
                         0357  VALIDATE_QUORUM_FILE:
                         0357
       0088 8F     BB    0357          PUSHR   #^M<R3,R7>                   ; Save CLUDCB
              7E   D4    035B          CLRL    -(SP)                        ; Assume invalid buffer
            0062   30    035D          BSBW    CALCULATE_CHECKSUM           ; Calculate quorum file checksum
              57   D5    0360          TSTL    R7                           ; Is checksum valid?
              11   12    0362          BNEQU   1$                           ; Br if not
              0C   29    0364          CMPC3   #CLUQF$S_IDENT,-             ; Validate ID area
              66         0366                  CLUQF$T_IDENT(R6),-
           0000'CF       0367                  CLUQF_IDENT_STRING
              09   12    036A          BNEQU   1$                           ; Br if invalid
              02   B1    036C          CMPW    #CLUQF$K_VERSION,-           ; Is version correct?
           0C A6         036E                  CLUQF$W_VERSION(R6)
              03   12    0370          BNEQU   1$                           ; Br if not
        6E    01   D0    0372          MOVL    #1,(SP)                      ; Indicate success
       0089 8F     BA    0375  1$:     POPR    #^M<R0,R3,R7>                ; Return status and restore register
              05         0379          RSB
```

```
                              037A  .SBTTL  CHECK_OWNER - Check quorum file ownership
                              037A  ;++
                              037A  ; CHECK_OWNER - Check quorum file ownership
                              037A  ;
                              037A  ; FUNCTIONAL DESCRIPTION:
                              037A  ;
                              037A  ;       This routine checks the quorum file owner block to see if it
                              037A  ;       is owned by a member of this nodes cluster.
                              037A  ;
                              037A  ; CALLING SEQUENCE:
                              037A  ;
                              037A  ;       JSB/BSBx CHECK_OWNER
                              037A  ;
                              037A  ; INPUTS:
                              037A  ;
                              037A  ;       R4 = address of CLUB
                              037A  ;       R6 = address of quorum file buffer
                              037A  ;
                              037A  ; OUTPUT:
                              037A  ;
                              037A  ;       R0 = Status
                              037A  ;           0 - Quorum file is owned by a foreign cluster
                              037A  ;           1 - Quorum file is owned by my cluster
                              037A  ;
                              037A  ;       R1-R2 Destroyed
                              037A  ;--
                              037A
                              037A  CHECK_OWNER:
                              037A
                  53    DD    037A          PUSHL   R3                              ; Save CLUDCB
                  7E    D4    037C          CLRL    -(SP)                           ; Assume foreign cluster
       14 A6   30 A4    D1    037E          CMPL    CLUB$Q_FTIME+4(R4), -           ; Same high order foundation times?
                              0383                  CLUQF$Q_FOU_TIME+4(R6)
                  3A    12    0383          BNEQU   1$                              ; Br if not
       10 A6   2C A4    D1    0385          CMPL    CLUB$Q_FTIME(R4), -             ; Same low order foundation times?
                              038A                  CLUQF$Q_FOU_TIME(R6)
                  33    12    038A          BNEQU   1$                              ; Br if not
                  06    29    038C          CMPC3   #CLUQF$S_FSYSID,-               ; Same founding system ID's?
               26 A4          038E                  CLUB$B_FSYSID(R4),-
               3E A6          0390                  CLUQF$B_FSYSID(R6)
                  28    12    0392          BNEQU   1$                              ; Br if not
          51   30 A6    3C    0394          MOVZWL  CLUQF$W_CSID_IDX(R6),R1         ; Get CSID index
00000000'GF       51    B1    0398          CMPW    R1,G^CLU$GW_MAXINDEX            ; Is index in range?
                  1E    1E    039F          BGEQU   1$                              ; Br if not
50   00000000'GF        D0    03A1          MOVL    G^CLU$GL_CLUSVEC,R0             ; Get vector address
          50    6041    D0    03A8          MOVL    (R0)[R1],R0                     ; Get entry (should be CSB address)
                  11    18    03AC          BGEQ    1$                              ; Br if no entry
               4C A0    D1    03AE          CMPL    CSB$L_CSID(R0),-                ; Do CSID's match?
               30 A6          03B1                  CLUQF$L_CSID(R6)
                  0A    12    03B3          BNEQ    1$                              ; Br if not
               38 A0    D1    03B5          CMPL    CSB$Q_SWINCARN(R0),-            ; Incarnation numbers match?
               28 A6          03B8                  CLUQF$Q_SWINCARN(R6)
                  03    12    03BA          BNEQU   1$                              ; Br if not
             6E   01    D0    03BC          MOVL    #1,(SP)                         ; Quorum file is owned by my cluster
                  09    BA    03BF  1$:     POPR    #^M<R0,R3>                      ; Restore CLUDCB
                        05    03C1          RSB
```

```
                    03C2  .SBTTL  CALCULATE_CHECKSUM - Calculate the quorum file checksum
                    03C2  ;++
                    03C2  ; CALCULATE_CHECKSUM - Calculate the quorum file checksum
                    03C2  ;
                    03C2  ; FUNCTIONAL DESCRIPTION:
                    03C2  ;
                    03C2  ;       This routine calulates the checksum of the quorum owner block
                    03C2  ;       pointed to by R6. It includes the field CLUQF$L_CHECKSUM in the
                    03C2  ;       checksum calculation.
                    03C2  ;
                    03C2  ; CALLING SEQUENCE:
                    03C2  ;
                    03C2  ;       JSB/BSBx CALCULATE_CHECKSUM
                    03C2  ;
                    03C2  ; INPUTS:
                    03C2  ;
                    03C2  ;       R6 = address of quorum file buffer
                    03C2  ;
                    03C2  ; OUTPUT:
                    03C2  ;
                    03C2  ;       R7 = Quorum block checksum
                    03C2  ;--
                    03C2
                    03C2  CALCULATE_CHECKSUM:
                    03C2
        0C    BB    03C2          PUSHR   #^M<R2,R3>                      ; Save registers
  52    12    D0    03C4          MOVL    #CLUQF$K_CHECK_LENGTH/4,R2      ; R2 = checksum longword count
  53    56    D0    03C7          MOVL    R6,R3                           ; Copy buffer address
        57    D4    03CA          CLRL    R7                              ; Form checksum in R7
  57    83    CC    03CC  1$:     XORL2   (R3)+,R7                        ; Accumulate checksum
     FA 52    F5    03CF          SOBGTR  R2,1$                           ; Br if more
        0C    BA    03D2          POPR    #^M<R2,R3>                      ; Restore registers
        05          03D4          RSB
```

```
                                03D5   .SBTTL  Quorum file error routines
                                03D5   ;++
                                03D5   ; QUORUM_DISK_TIMEOUT - Quorum disk timeout
                                03D5   ; QUORUM_FILE_ERROR - Quorum file error
                                03D5   ;
                                03D5   ; FUNCTIONAL DESCRIPTION:
                                03D5   ;
                                03D5   ;       This routine handles timeouts and other errors associated
                                03D5   ;       with the quorum disk.
                                03D5   ;
                                03D5   ; CALLING SEQUENCE:
                                03D5   ;
                                03D5   ;       JSB/BSBx QUORUM_DISK_TIMEOUT
                                03D5   ;       JSB/BSBx QUORUM_FILE_ERROR
                                03D5   ;
                                03D5   ; INPUTS:
                                03D5   ;
                                03D5   ;       R0 = address of error message
                                03D5   ;       R3 = address of CLUDCB
                                03D5   ;       R4 = address of CLUB
                                03D5   ;
                                03D5   ; OUTPUT:
                                03D5   ;
                                03D5   ;       R0-R2 destroyed
                                03D5   ;--
                                03D5           .ENABLE LSB
                                03D5
                                03D5   QUORUM_DISK_TIMEOUT:
                                03D5   QUORUM_FILE_RETRY:
                                03D5
               51    02   B0    03D5           MOVW    #CLUDCB$M_QS_READY,R1           ; The new state is READY
                     0B   11    03D8           BRB     1$
                                03DA
                                03DA   QUORUM_FILE_ERROR:
                                03DA
                     50   DD    03DA           PUSHL   R0                              ; Save error message address
                   0035   30    03DC           BSBW    REQUEST_CSP                     ; Request the CSP process
               50    8E   D0    03DF           MOVL    (SP)+,R0                        ; Restore error message address
               51    01   B0    03E2           MOVW    #CLUDCB$M_QS_NOT_READY,R1       ; The new state is not ready
                     20   BB    03E5   1$:     PUSHR   #^M<R5>                         ; Save R5
                     55   D4    03E7           CLRL    R5                              ; No CSB (input to CNX$CONFIG_CHANGE
            7E    20 A3   3C    03E9           MOVZWL  CLUDCB$W_STATE(R3),-(SP)        ; Save current state
            20 A3    51   B0    03ED           MOVW    R1,CLUDCB$W_STATE(R3)           ; Update state
                     03   E2    03F1           BBSS    #CLUDCB$V_QF_ERROR,-            ; Br if an error has already been re
               03 22 A3         03F3                   CLUDCB$W_FLAGS(R3),2$
                  FC07'   30    03F6           BSBW    CNX$CONFIG_CHANGE               ; Output error message
               8E    03   D3    03F9   2$:     BITL    #<CLUDCB$M_QS_NOT_READY! -      ; Was state NOT_READY or READY?
                                03FC                     CLUDCB$M_QS_READY>,(SP)+
                     13   12    03FC           BNEQU   3$                              ; Br if yes
                          CA    03FE           BICL    #<CLUB$M_QF_ACTIVE! -           ; Clear the CLUB bits
                                03FF                     CLUB$M_QF_DYNVOTE! -
                                03FF                     CLUB$M_QF_FAILED_NODE>,-
   1C A4    41000002 8F         03FF                   CLUB$L_FLAGS(R4)
       50    0000'CF      9E    0406           MOVAB   W^QDDISCON_MSG,R0               ; Point to quorum disk disconnect me
                  FBF2'   30    040B           BSBW    CNX$CONFIG_CHANGE               ; Output message
                  FBEF'   30    040E           BSBW    CNX$DISK_CHANGE                 ; Let connection manager know
                     20   BA    0411   3$:     POPR    #^M<R5>                         ; Restore R5
```

```
                                  05  0413   941          RSB
                                      0414   942
                                      0414   943          .DISABLE LSB
```

```
                    0414  .SBTTL  REQUEST_CSP - Request the CSP process
                    0414  ;++
                    0414  ; REQUEST_CSP - Request the CSP process
                    0414  ;
                    0414  ; FUNCTIONAL DESCRIPTION:
                    0414  ;
                    0414  ;       If it has not already been requested, this routine requests the
                    0414  ;       quorum thread of the CSP process.
                    0414  ;
                    0414  ; CALLING SEQUENCE:
                    0414  ;
                    0414  ;       JSB/BSBx REQUEST_CSP
                    0414  ;
                    0414  ; INPUTS:
                    0414  ;
                    0414  ;       R3 = address of CLUDCB
                    0414  ;
                    0414  ; OUTPUT:
                    0414  ;
                    0414  ;       R0-R2 destroyed
                    0414  ;--
                    0414
                    0414  REQUEST_CSP:
          18   BB   0414          PUSHR   #^M<R3,R4>                      ; Save CLUDCB and CLUB pointers
     51   07   D0   0416          MOVL    #CSP$_LOCAL,R1                  ; Send to local CSP
          52   D4   0419          CLRL    R2                              ; No CSD pointer
          53   D4   041B          CLRL    R3                              ; Must be zero
     54   07   D0   041D          MOVL    #CSD$K_QUORUM,R4                ; R4 = client code
        FBDD'  30   0420          BSBW    EXE$CSP_COMMAND                 ; Request CSP
          18   BA   0423          POPR    #^M<R3,R4>                      ; Restore CLUDCB and CLUB pointers
               05   0425          RSB
                    0426
```

```
                         0426  .SBTTL  CHECK_ERROR - Check to see if error is fatal
                         0426  ;++
                         0426  ; CHECK_ERROR - Check to see if error is fatal
                         0426  ;
                         0426  ; FUNCTIONAL DESCRIPTION:
                         0426  ;
                         0426  ;       This routine checks the error status to see if we should simply retry.
                         0426  ;       We then cause a cluster state change and also cause mount verification
                         0426  ;       to be invoked.  This is necessary because the "internal" IRP
                         0426  ;       format used by quorum I/Os does not trigger mount verification.
                         0426  ;
                         0426  ;       In the case of accidental write-lock, quorum I/O is retried.
                         0426  ;
                         0426  ; CALLING SEQUENCE:
                         0426  ;
                         0426  ;       JSB/BSBx CHECK_ERROR
                         0426  ;
                         0426  ; INPUTS:
                         0426  ;
                         0426  ;       R3 = address of CLUDCB
                         0426  ;       R4 = address of CLUB
                         0426  ;       R5 = address of UCB
                         0426  ;
                         0426  ; OUTPUT:
                         0426  ;
                         0426  ;       R0 = Status (low bit)
                         0426  ;           0 - no recovery - normal error processing
                         0426  ;           1 - non-fatal error
                         0426  ;
                         0426  ;--
                         0426
                         0426 CHECK_ERROR:
                         0426
              3E    BB   0426          PUSHR   #^M<R1,R2,R3,R4,R5>
                         0428  ;
   51    38 A5      3C   0428          MOVZWL  IRP$L_IOST1(R5),R1      ; Get the error status
                         042C          ;
                         042C          ; If the medium is offline, or the volume is
                         042C          ; invalid, the error can be recovered from.
                         042C          ;
   51   0000'8F     B1   042C          CMPW    #SS$_MEDOFL,R1          ; Is the media (disk volume) offline?
              37    13   0431          BEQL    40$                     ; Branch if true
   51   0000'8F     B1   0433          CMPW    #SS$_VOLINV,R1          ; Is the volume invalid?
              30    13   0438          BEQL    40$                     ; Branch if true
                         043A          ;
                         043A          ; If the volume has been writelocked, make sure that it was
                         043A          ; an accidental writelock.  If the software writelock bit is
                         043A          ; on, then the volume was mounted with the volume write protected.
                         043A          ; If the bit is not set, then the volume has been mounted for
                         043A          ; read/write access, and has since been (accidentally) write protected.
                         043A          ; The first time through this code and any time we are in the cluster or
                         043A          ; vote states, we put everything in mount verification and cause a
                         043A          ; cluster state change and return to the active state.  All other times,
                         043A          ; we remain in the same state and quietly return.  This saves many
                         043A          ; trees.
                         043A          ;
   51   0000'8F     B1   043A          CMPW    #SS$_WRITLCK,R1         ; Is the device writelocked?
```

```
                   05   13  043F          BEQL    10$
          50       51   D0  0441          MOVL    R1,R0                       ; Get an error code in R0
                   21   11  0444          BRB     30$                         ; Go back to treat it as real error
         00000000'8F   E0  0446  10$:    BBS     #DEV$V_SWL,-                ; Branch if software writelocked
              18 38 A5      044C                  UCB$L_DEVCHAR(R5),30$
                   06   E3  044F          BBCS    #CLUDCB$V_QF_WRL_ERR,-      ; See if this is the first time
              08 22 A3      0451                  CLUDCB$W_FLAGS(R3),15$
                 24 A3  94  0454          CLRB    CLUDCB$B_COUNTER(R3)        ; Restart counter in case in cluster state
                   04   E1  0457          BBC     #CLUDCB$V_QS_VOTE,-         ; Is it a dangerous state
              08 20 A3      0459                  CLUDCB$W_STATE(R3),25$      ; No - leave it there
        50    0000'CF   9E  045C  15$:    MOVAB   W^QDWRLERROR_MSG,R0         ; Point to write error message
                 FF71   30  0461  20$:    BSBW    QUORUM_FILE_RETRY           ; Go try again
          50       01   D0  0464  25$:    MOVL    #1,R0                       ; Error recovery in progress
                   3E   BA  0467  30$:    POPR    #^M<R1,R2,R3,R4,R5>
                        05  0469          RSB
                            046A  ;
         00000000'GF   16  046A  40$:    JSB     G^EXE$CLUTRANIO             ; Get everyting in mount verification
                   EF   11  0470          BRB     20$
                            0472
                            0472
                            0472          .END
```

```
BUILD_QUORUM_FILE                000001F5 R     04    CLUDCB$W_STATE              = 00000020
CALCULATE_CHECKSUM               000003C2 R     04    CLUQF$B_FSYSID              = 0000C03E
CHECK_ERROR                      00000426 R     04    CLUQF$B_IGNORE              = 00000048
CHECK_OWNER                      0000037A R     04    CLUQF$K_ACT_LENGTH          = 00000004
CLU$GB_QDISK                     ********    X  03    CLUQF$K_CHECK_LENGTH        = 00000048
CLU$GL_CLUB                      ********    X  03    CLUQF$K_LENGTH              = 00000204
CLU$GL_CLUSVEC                   ********    X  04    CLUQF$K_VERSION             = 00000002
CLU$GW_MAXINDEX                  ********    X  04    CLUQF$L_ACT_COUNT           = 00000200
CLU$GW_QDSKINTERVAL              ********    X  03    CLUQF$L_CSID                = 00000030
CLUB$B_FSYSID                  = 00000026             CLUQF$M_QUORUM              = 00000001
CLUB$L_CLUDCB                  = 000000B4             CLUQF$Q_FOU_TIME            = 00000010
CLUB$L_FLAGS                   = 0000001C             CLUQF$Q_SWINCARN            = 00000028
CLUB$L_FOREIGN_CLUSTER         = 000000C8             CLUQF$S_FSYSID              = 00000006
CLUB$L_LOCAL_CSID              = 00000060             CLUQF$S_IDENT               = 0000000C
CLUB$M_QF_ACTIVE               = 00000002             CLUQF$T_IDENT               = 00000000
CLUB$M_QF_DYNVOTE              = 40000000             CLUQF$V_QUORUM              = 00000000
CLUB$M_QF_FAILED_NODE          = 01000000             CLUQF$W_CSID_IDX            = 00000030
CLUB$Q_FTIME                   = 0000002C             CLUQF$W_FLAGS               = 0000000E
CLUB$Q_LST_TIME                = 0000003C             CLUQF$W_QUORUM              = 00000034
CLUB$V_CLUSTER                 = 00000000             CLUQF$W_VERSION             = 0000000C
CLUB$V_QF_FAILED_NODE          = 00000018             CLUQF$W_VOTES               = 00000036
CLUB$V_QUORUM                  = 0000001C             CLUQF_IDENT_STRING            00000000 R     02
CLUB$W_QUORUM                  = 00000020             CNX$BUGCHECK_CLUSTER          ********    X  04
CLUB$W_VOTES                   = 00000022             CNX$CONFIG_CHANGE             ********    X  04
CLUDCB$B_COUNTER               = 00000024             CNX$DISK_CHANGE               ********    X  04
CLUDCB$B_SUBTYPE               = 0000000B             CNX$QUORUM_INIT               00000000 RG    03
CLUDCB$B_TYPE                  = 0000000A             CSB$L_CSID                  = 0000004C
CLUDCB$K_LENGTH                = 00000229             CSB$Q_SWINCARN              = 00000038
CLUDCB$L_ACT_COUNT             = 00000018             CSD$K_QUORUM                = 00000007
CLUDCB$L_IRP                   = 00000010             CSP$_LOCAL                  = 00000007
CLUDCB$L_QFLBN                 = 0000001C             CYCLE_COUNT                 = 00000002
CLUDCB$L_TQE                   = 00000014             DEV$V_SWL                     ********    X  04
CLUDCB$L_UCB                   = 0000000C             DISPATCH                      0000011F R     04
CLUDCB$M_QF_ERROR              = 00000008             DYN$C_CLU                   = 00000065
CLUDCB$M_QF_RIP                = 00000002             DYN$C_CLU_CLUDCB            = 00000005
CLUDCB$M_QF_TIM                = 00000001             DYN$C_IRP                   = 0000000A
CLUDCB$M_QF_WIP                = 00000004             DYN$C_TQE                   = 0000000F
CLUDCB$M_QF_WRL_ERR            = 00000040             EXE$ALONONPAGED               ********    X  03
CLUDCB$M_QS_ACTIVE             = 00000004             EXE$CLUTRANIO                 ********    X  04
CLUDCB$M_QS_CLUSTER            = 00000008             EXE$CSP_COMMAND               ********    X  04
CLUDCB$M_QS_NOT_READY          = 00000001             EXE$GQ_SYSTIME                ********    X  04
CLUDCB$M_QS_READY              = 00000002             EXE$INSIOQ                    ********    X  04
CLUDCB$M_QS_VOTE               = 00000010             IO$_READPBLK                = 0000000C
CLUDCB$S_BUFFER                = 00000204             IO$_WRITEPBLK               = 0000000B
CLUDCB$S_DISK_QUORUM           = 00000010             IOC$CVTLOGPHY                 ********    X  04
CLUDCB$T_BUFFER                = 00000025             IPL$_IOPOST                 = 00000004
CLUDCB$V_QF_ERROR              = 00000003             IPL$_SCS                    = 00000008
CLUDCB$V_QF_FIRST_ERR          = 00000005             IPL$_SYNCH                  = 00000008
CLUDCB$V_QF_RIP                = 00000001             IPL$_TIMER                  = 00000008
CLUDCB$V_QF_TIM                = 00000000             IRP$B_PRI                   = 00000023
CLUDCB$V_QF_WIP                = 00000002             IRP$B_TYPE                  = 0000000A
CLUDCB$V_QF_WRL_ERR            = 00000006             IRP$K_LENGTH                = 000000C4
CLUDCB$V_QS_NOT_READY          = 00000000             IRP$L_BCNT                  = 00000032
CLUDCB$V_QS_READY              = 00000001             IRP$L_IOST1                 = 00000038
CLUDCB$V_QS_VOTE               = 00000004             IRP$L_PID                   = 0000000C
CLUDCB$W_FLAGS                 = 00000022             IRP$L_SVAPTE                = 0000002C
CLUDCB$W_SIZE                  = 00000008             IRP$L_UCB                   = 0000001C
```

```
IRP$M_PHYSIO                    = 00000100
IRP$W_BOFF                      = 00000030
IRP$W_FUNC                      = 00000020
IRP$W_SIZE                      = 00000008
IRP$W_STS                       = 0000002A
MMG$GL_SPTBASE                    ********   X     04
PR$_IPL                           ********   X     04
QDCON_MSG                         ********   X     04
QDDISCON_MSG                      ********   X     04
QDFORCLUS_MSG                     ********   X     04
QDINVDAT_MSG                      ********   X     04
QDRDERROR_MSG                     ********   X     04
QDTIMOUT_MSG                      ********   X     04
QDWRERROR_MSG                     ********   X     04
QDWRLERROR_MSG                    ********   X     04
QUORUM_DISK_TIMEOUT               000003D5 R       04
QUORUM_FILE_ERROR                 000003DA R       04
QUORUM_FILE_RETRY                 000003D5 R       04
QUORUM_TIMEOUT                    00000000 RG      04
READ_COMPLETE                     00000097 RG      04
READ_COMPLETE_ACTIVE              0000016E R       04
READ_COMPLETE_CLUSTER             000001A6 R       04
READ_COMPLETE_READY               0000012F R       04
READ_COMPLETE_VOTE                000001A6 R       04
READ_QUORUM_FILE                  00000039 R       04
REQUEST_CSP                       00000414 R       04
SB$B_SYSTEMID                   = 00000018
SB$Q_SWINCARN                   = 0000002C
SB$S_SYSTEMID                   = 00000006
SCS$GA_LOCALSB                    ********   X     04
SS$_MEDOFL                        ********   X     04
SS$_NORMAL                        ********   X     03
SS$_VOLINV                        ********   X     04
SS$_WRITLCK                       ********   X     04
TQE$B_RQTYPE                    = 0000000B
TQE$B_TYPE                      = 0000000A
TQE$C_SSREPT                    = 00000005
TQE$K_LENGTH                    = 00000030
TQE$L_FPC                       = 0000000C
TQE$L_FR3                       = 00000010
TQE$L_FR4                       = 00000014
TQE$Q_DELTA                     = 00000020
TQE$W_SIZE                      = 00000008
UCB$L_DEVCHAR                   = 00000038
UCB$V_NOCNVRT                   = 00000002
UCB$W_DEVSTS                    = 00000068
UCB$W_QLEN                      = 0000006A
VA$M_BYTE                       = 000001FF
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VALIDATE_QUORUM_FILE              00000357 R       04
WRITE_COMPLETE                    000002DA RG      04
WRITE_QUORUM_ACT                  0000026C R       04
WRITE_QUORUM_OWNACT               0000025F R       04
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                 Allocation             PSECT No.  Attributes
----------                 ----------             ---------  ----------
.  ABS  .                  00000000  (      0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                      00000000  (      0.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$060                     0000000C  (     12.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$002                     000000C7  (    199.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$100                     00000472  (   1138.)  04 (   4.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                    Page faults    CPU Time        Elapsed Time
-----                    -----------    --------        ------------
Initialization                    35    00:00:00.05     00:00:02.35
Command processing               137    00:00:00.46     00:00:03.48
Pass 1                           420    00:00:10.39     00:00:36.94
Symbol table sort                  0    00:00:01.64     00:00:07.11
Pass 2                           188    00:00:02.44     00:00:10.00
Symbol table output               20    00:00:00.11     00:00:00.11
Psect synopsis output              3    00:00:00.02     00:00:00.51
Cross-reference output             0    00:00:00.00     00:00:00.00
Assembler run totals             805    00:00:15.11     00:01:00.50

The working set limit was 1950 pages.
90025 bytes (176 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1566 non-local and 44 local symbols.
1055 source lines were read in Pass 1, producing 21 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

                                          +--------------------------+
                                          ! Macro library statistics !
                                          +--------------------------+

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1                 3
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       11
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     5
TOTALS (all libraries)                               19

1637 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:QUORUM/OBJ=OBJ$:QUORUM MSRC$:QUORUM/UPDATE=(ENH$:QUORUM)+EXECML$/LIB+LIB$:CLUSTER/LIB
```

0398 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
LIS

REBLDLOCK
LIS